ETUDE Engine Documentation

Paul M. Heider & the TBIC Team

Sep 09, 2021

HIgh-Level Content:

1 Documentation **3**

2 Sample Runs **5**
2.1 Basic Run . . . 5
2.2 Specifying Annotation Configs . . . 6
2.3 Scoring on Different Fields . . . 8

3 Custom Evaluation Print-Outs **11**
3.1 Contextually-Grounded Annotation Examples . . . 12

4 Configuring Annotation Extraction **15**

5 Dependencies **17**

6 Building with PyInstaller **19**

7 Testing **21**

8 API Documentation **23**
8.1 args_and_configs.py Functions . . . 23
8.2 etude.py Functions . . . 24
8.3 scoring_metrics.py Functions . . . 24
8.4 text_extraction.py Functions . . . 26

9 Configuration Files **29**
9.1 Score Keys . . . 29
9.2 Score Values . . . 29

10 Input Formats & Support Details **31**
10.1 Simple Plain Text . . . 31
10.2 Structured Plain Text (e.g., csv) . . . 31
10.3 XML Formats . . . 32

11 Evaluating Matches **33**
11.1 Exact Match . . . 33
11.2 Partial Match . . . 33
11.3 Fully-Contained Match . . . 33
11.4 Start Match . . . 34

11.5 End Match 34
11.6 Doc-Property Match 34

12 Evaluating Normalization **35**

13 Evaluating Context Attributes **37**
13.1 Sample Test Data 37

14 Output Formats **39**

15 Indices and tables **41**

Python Module Index **43**

Index **45**

CHAPTER 1

Documentation

The latest documentation (compiled from the contents of the *docs* folder) can be viewed on-line: ETUDE Engine's documentation

Documentation for the ETUDE engine is managed via reStructuredText files and Sphinx. If you don't have Sphinx installed, you should check out a quick primer (First Steps with Sphinx) or install it as below:

```
## If you don't have Sphinx installed already
pip install Sphinx

## Generate a locally viewable HTML version
cd docs
make html
```

The latest version of the documentation can be generated as locally viewable HTML: file:///path/to/git/repository/docs/_build/html/index.html

CHAPTER 2

Sample Runs

2.1 Basic Run

The simplest test run requires that we specify a reference directory and a test directory. The default file matching assumes that our reference and test files match names exactly and both end in *.xml*. With just the two directory arguments, we get micro-average scores for the default metrics across the full directory.

```
python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test
```

exact	TP	FP	TN	FN
micro-average	374.0	8.0	0.0	108.0

Note: You may get a warning if you run the previous command from a directory other than *$ETUDE_DIR*:

```
ERROR: Config file is missing or unreadable:  config/i2b2_2016_track-1.conf
ERROR: No reference patterns extracted from config. Bailing out now.
```

This warning is because the default configuration files use relative paths. See the section below

In the next sample runs, you can see how to include a per-file score breakdown and a per-annotation-type score breakdown.

```
python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test \
    --by-file
```

exact	TP	FP	TN	FN
micro-average	374.0	8.0	0.0	108.0
0005_gs.xml	36.0	0.0	0.0	0.0
0016_gs.xml	23.0	0.0	0.0	31.0
0267_gs.xml	29.0	0.0	0.0	34.0
0273_gs.xml	0.0	0.0	0.0	35.0
0389_gs.xml	32.0	8.0	0.0	8.0
0475_gs.xml	46.0	0.0	0.0	0.0
0617_gs.xml	38.0	0.0	0.0	0.0
0709_gs.xml	45.0	0.0	0.0	0.0
0982_gs.xml	100.0	0.0	0.0	0.0
0992_gs.xml	25.0	0.0	0.0	0.0
macro-average by file	374.0	8.0	0.0	108.0

```
python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test \
    --by-type
```

exact	TP	FP	TN	FN
micro-average	374.0	8.0	0.0	108.0
Age	63.0	2.0	0.0	29.0
DateTime	91.0	2.0	0.0	33.0
HCUnit	61.0	4.0	0.0	15.0
OtherGeo	1.0	0.0	0.0	4.0
OtherID	7.0	0.0	0.0	0.0
OtherOrg	18.0	0.0	0.0	3.0
Patient	16.0	0.0	0.0	3.0
PhoneFax	5.0	0.0	0.0	1.0
Provider	54.0	0.0	0.0	10.0
SSN	0.0	0.0	0.0	0.0
StateCountry	24.0	0.0	0.0	9.0
StreetCity	28.0	0.0	0.0	1.0
Zip	4.0	0.0	0.0	0.0
eAddress	2.0	0.0	0.0	0.0
macro-average by type	374.0	8.0	0.0	108.0

2.2 Specifying Annotation Configs

We can use the same reference corpus to analyze annotations generated by UIMA's DateTime tutorial (see link below). A minimal run requires creating a matching dataset for the default configurations. Process the I2B2 dev set using the DateTime tutorial provided with UIMA. Then, because the output files for the I2B2 dev-annotations end in *.xml* but the UIMA tutorial files end in *.txt*, you need to specify a file suffix translation rule. Also, the annotations are encoded slightly differently by the tutorial descriptor than by the I2B2 reference. As such, you will need to load a different configuration for the test directory to tell ETUDE how to find and extract the annotations.

Link: http://uima.apache.org/downloads/releaseDocs/2.2.2-incubating/docs/html/tutorials_and_users_guides/tutorials_and_users_guides.html#ugr.tug.aae.building_aggregates

```
export I2B2_CORPUS="/path/to/Corpora and annotations/2016 NGRID challenge (deid)/2016_
↪track_1-deidentification"

export I2B2_OUTPUT="/tmp/datetime-out"
mkdir $I2B2_OUTPUT

$UIMA_HOME/bin/runAE.sh \
  $UIMA_HOME/examples/descriptors/tutorial/ex3/TutorialDateTime.xml \
  $I2B2_CORPUS/dev-text \
  $I2B2_OUTPUT

python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $I2B2_OUTPUT \
    --by-type \
    --file-suffix ".xml" ".txt" \
    --test-config config/CAS_XMI.conf
```

exact	TP	FP	TN	FN
micro-average	0.0	39.0	0.0	124.0
DateTime	0.0	39.0	0.0	124.0
macro-average	0.0	39.0	0.0	124.0

You may be surprised that the UIMA tutorial doesn't seem to get a single DateTime annotation correct. When you look at the annotations, they seem correct. This discrepancy arises because the default matching style is "exact" matching. This means that the character offset spans needs to be identical. You can select a different matching style with the *–fuzzy-match-flags* option. Using *partial* matching instead shows True Positives (TPs) for all spans that at least partially overlap. See the more thorough coverage on the various options under Evaluating Matches in our documentation.

```
python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $I2B2_OUTPUT \
    --by-type \
    --file-suffix ".xml" ".txt" \
    --test-config config/CAS_XMI.conf \
    --fuzzy-match-flags partial
```

exact	TP	FP	TN	FN
micro-average	39.0	0.0	0.0	85.0
DateTime	39.0	0.0	0.0	85.0
macro-average	39.0	0.0	0.0	85.0

If you run this example without the *-–test-config* argument, you should see all FN matches because nothing can be extracted from the test corpus.

```
python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $I2B2_OUTPUT \
    --file-suffix ".xml" ".txt"
```

exact	TP	FP	TN	FN
micro-average	0.0	0.0	0.0	482.0

2.3 Scoring on Different Fields

The above examples show scoring based on the default key in the configuration file used for matching the reference to the test configuration. You may wish to group annotations on different fields, such as the parent class or long description. See the more thorough coverage on the various options under Configuration Files in our documentation.

```
## You can see output for this command above
python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test \
    --by-type

python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test \
    --by-type \
    --score-key "Parent"

python $ETUDE_DIR/etude.py \
    --reference-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_reference \
    --test-input $ETUDE_DIR/tests/data/i2b2_2016_track-1_test \
    --by-type \
    --score-key "i2b2 14/16"
```

exact	TP	FP	TN	FN
micro-average	375.0	7.0	0.0	107.0
Address	56.0	0.0	0.0	10.0
Contact Information	7.0	0.0	0.0	1.0
Identifiers	7.0	0.0	0.0	0.0
Locations	80.0	4.0	0.0	22.0
Names	70.0	0.0	0.0	13.0
Time	155.0	3.0	0.0	61.0
macro-average by type	375.0	7.0	0.0	107.0

exact	TP	FP	TN	FN
micro-average	374.0	8.0	0.0	108.0
ACCOUNT	0.0	0.0	0.0	0.0
AGE	63.0	2.0	0.0	29.0
BIOID	0.0	0.0	0.0	0.0
CITY	24.0	0.0	0.0	1.0
COUNTRY	14.0	0.0	0.0	7.0
DATE	91.0	2.0	0.0	33.0
DEVICE	0.0	0.0	0.0	0.0
DOCTOR	54.0	0.0	0.0	10.0
EMAIL	0.0	0.0	0.0	0.0
FAX	0.0	0.0	0.0	0.0
HEALTHPLAN	0.0	0.0	0.0	0.0
HOSPITAL	61.0	4.0	0.0	15.0
IDNUM	0.0	0.0	0.0	0.0
IPADDRESS	0.0	0.0	0.0	0.0
LICENSE	7.0	0.0	0.0	0.0
LOCATION-OTHER	1.0	0.0	0.0	4.0
MEDICALRECORD	0.0	0.0	0.0	0.0
ORGANIZATION	18.0	0.0	0.0	3.0
PATIENT	16.0	0.0	0.0	3.0
PHONE	5.0	0.0	0.0	1.0
SSN	0.0	0.0	0.0	0.0
STATE	10.0	0.0	0.0	2.0
STREET	4.0	0.0	0.0	0.0
URL	2.0	0.0	0.0	0.0
USERNAME	0.0	0.0	0.0	0.0
VEHICLE	0.0	0.0	0.0	0.0
ZIP	4.0	0.0	0.0	0.0
macro-average by type	374.0	8.0	0.0	108.0

CHAPTER 3

Custom Evaluation Print-Outs

The majority of you evaluation output customization can be handled by the above command-line arguments. However, sometimes you'll need to generate output that exactly matches some very specific formatting requirements. For these instances, ETUDE supports custom print functions. Currently, those print functions must be hard-coded into *scoring_metrics.py*. Our roadmap includes the ability to load and trigger these print functions from a standard folder to make the system much more modular. Until that point, you can see an example custom print-out that targets the 2018 n2c2 Track 1 output format. The configurations for this sample are in our sister repository: ETUDE Engine Configs for n2c2 The original evaluation script for the competition, used as a point of reference, can be found on github: Evaluation scripts for the 2018 N2C2 shared tasks on clinical NLP See the more thorough coverage on the various Output Formats in our documentation.

```
export ETUDE_DIR=etude-engine
export ETUDE_CONFIGS_DIR=etude-engine-configs

export N2C2_DATA=/tmp/n2c2

python ${ETUDE_DIR}/etude.py \
  --reference-input ${N2C2_DATA}/train_annotations \
   --reference-config ${ETUDE_CONFIGS_DIR}/n2c2/2018_n2c2_track-1.conf \
   --test-input ${N2C2_DATA}/train_annotations \
   --test-config ${ETUDE_CONFIGS_DIR}/n2c2/2018_n2c2_track-1.conf \
   --no-metrics \
   --print-custom "2018 n2c2 track 1" \
   --fuzzy-match-flag exact \
   --file-suffix ".xml" \
   --empty-value 0.0


****************************************** TRACK 1␣
↪******************************************
                     ------------ met -------------    ------ not met -------    --␣
↪overall ---
                     Prec.   Rec.    Speci.  F(b=1)    Prec.   Rec.    F(b=1)    ␣
↪F(b=1)  AUC
          Abdominal  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
```

(continues on next page)

(continued from previous page)

```
        Advanced-cad  1.0000  1.0000  0.0000  1.0000    0.0000  0.0000  0.0000    0.
↪5000  0.5000
       Alcohol-abuse  0.0000  0.0000  1.0000  0.0000    1.0000  1.0000  1.0000    0.
↪5000  0.5000
         Asp-for-mi  1.0000  1.0000  0.0000  1.0000    0.0000  0.0000  0.0000    0.
↪5000  0.5000
         Creatinine  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
       Dietsupp-2mos  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
         Drug-abuse  0.0000  0.0000  1.0000  0.0000    1.0000  1.0000  1.0000    0.
↪5000  0.5000
            English  1.0000  1.0000  0.0000  1.0000    0.0000  0.0000  0.0000    0.
↪5000  0.5000
              Hba1c  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
           Keto-1yr  0.0000  0.0000  1.0000  0.0000    1.0000  1.0000  1.0000    0.
↪5000  0.5000
      Major-diabetes  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
     Makes-decisions  1.0000  1.0000  0.0000  1.0000    0.0000  0.0000  0.0000    0.
↪5000  0.5000
            Mi-6mos  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
                     ------------------------------    ----------------------    ----
↪----------
     Overall (micro)  1.0000  1.0000  1.0000  1.0000    1.0000  1.0000  1.0000    1.
↪0000  1.0000
     Overall (macro)  0.7692  0.7692  0.6923  0.7692    0.6923  0.6923  0.6923    0.
↪7308  0.7308

                                                  10 files found
```

3.1 Contextually-Grounded Annotation Examples

A second class of custom outputs is to generate listings of real annotations with left- and right-margins of context. Most often, you will want to use this type of output to generate a listing of all the FP annotations your system generated or all the FN annotations your system failed to find.

The generation of this output is dependent on a score card having been written to disk during a normal evaluation run. You'll also want to make sure to have generated a system output directory. Both flags are show in examples below. Additional flags let you determine how much of a context window (in characters) you want to see on the left and right of the annotation.

If we focus solely on the *partial* matches, then we're guaranteed to get FP and FN annotations that don't overlap. We don't distinguish between span mismatches and type mismatches.

```
export ETUDE_DIR=etude-engine

python3 ${ETUDE_DIR}/etude.py \
  --reference-input ${ETUDE_DIR}/tests/data/i2b2_2016_track-1_reference \
  --reference-config ${ETUDE_DIR}/config/i2b2_2016_track-1.conf \
  --test-input ${ETUDE_DIR}/tests/data/i2b2_2016_track-1_test \
  --test-config ${ETUDE_DIR}/config/i2b2_2016_track-1.conf \
```

(continues on next page)

(continued from previous page)

```
  --file-suffix "xml" \
  --by-type \
  -m FP FN \
  --fuzzy-match-flags partial \
  --pretty-print \
  --test-out /tmp/system \
  --write-score-cards

## Use standard settings
python3 ${ETUDE_DIR}/extract_samples.py \
  --score-card /tmp/system/metrics_partial_score_card.csv \
  --annotation-out /tmp/system

## Show a larger left margin than right margin
python3 ${ETUDE_DIR}/extract_samples.py \
  --score-card /tmp/system/metrics_partial_score_card.csv \
  --annotation-out /tmp/system \
  --left-margin 25 \
  --right-margin 10

## Only print the FP annotations
python3 ${ETUDE_DIR}/extract_samples.py \
  --score-card /tmp/system/metrics_partial_score_card.csv \
  --annotation-out /tmp/system \
  --metrics FP

## The system output filenames differ from the reference
## filenames in that they end in '.txt.xmi' rather than
## just '.txt'
python3 ${ETUDE_DIR}/extract_samples.py \
  --score-card /tmp/system/metrics_partial_score_card.csv \
  --annotation-out /tmp/system \
  --file-suffix ".txt" ".txt.xmi"
```

CHAPTER 4

Configuring Annotation Extraction

Several sample configurations are provided in the config/ folder. Each long name for an annotation description should be unique due to how Python's configuration parser works. XPath's should also be unique within a config file but do not programmitically need to be. The begin and end attribute are required for a pattern to be scorable.

```
[ Long Name or Description ]
Parent:          (optional; useful for merging multiple child types together for␣
↪scoring)
Short Name:      (optional; useful for displaying as column output name and merging
                   multiple XPaths into a single scoring category)
XPath:           (required for XML; pattern used by XPath to find annotation)
Begin Attr:      (required; beginning or start offset attribute name)
End Attr:        (required; end offset attribute name)
Text Attr:       (optional; not used by anything currently)
```

Additional interesting or useful configuration files can be found in our sister repository: ETUDE Engine Configs

CHAPTER 5

Dependencies

Python module requirements for running ETUDE are included in the requirements.txt file. You should be able to install all non-default packages using pip:

```
pip install -r requirements
```

CHAPTER 6

Building with PyInstaller

After installing all required dependencies (as above), you can opt to create a stand-alone version of the ETUDE engine with PyInstaller.

The vanilla creation is .. code:: bash

    cd $ETUDE_ENGINE_DIR

    pyinstaller –onefile –distpath=dist/linux etude.py pyinstaller –onefile –distpath=dist/osx etude.py pyinstaller –onefile –distpath=dist/windows etude.py

CHAPTER 7

Testing

Unit testing is done with the pytest module. Because of a bug in how tests are processed in Python, you should run pytest indirectly rather than directly:

```
python -m pytest tests/

## You can also generate a coverate report in html format
python3.7 -m pytest --cov-report html:cov_html_py3.7 --cov=./ tests/

## The junit file is helpful for automated systems or CI pipelines
python -m pytest --junitxml=junit.xml tests
```

CHAPTER 8

API Documentation

8.1 args_and_configs.py Functions

args_and_configs.**align_patterns**(*reference_patterns*, *test_patterns*, *collapse_all_patterns*)

args_and_configs.**extract_brat_patterns**(*annotations*, *config*, *sect*, *display_name*, *key_value*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

args_and_configs.**extract_delimited_patterns**(*annotations*, *config*, *sect*, *display_name*, *key_value*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

args_and_configs.**extract_document_data**(*document_data*, *config*, *sect*)

Add handling for any special document-level data fields

args_and_configs.**extract_json_patterns**(*annotations*, *config*, *sect*, *display_name*, *key_value*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

args_and_configs.**extract_namespaces**(*namespaces*, *config*, *sect*)

args_and_configs.**extract_patterns**(*annotations*, *config*, *sect*, *score_key*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

Iterates over each config section not handled by extract_namespaces() or extract_document_data() and pulls out the pattern-level configuration details.

args_and_configs.**extract_semeval_patterns**(*annotations*, *config*, *sect*, *display_name*, *key_value*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

args_and_configs.**extract_xpath_patterns**(*annotations*, *config*, *sect*, *display_name*, *key_value*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

args_and_configs.**extract_xpath_spanless_patterns**(*annotations*, *config*, *sect*, *display_name*, *key_value*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

args_and_configs.**get_arguments**(*command_line_args*)

args_and_configs.**initialize_arg_parser**()

args_and_configs.**process_config**(*config_file*, *score_key*, *score_values*, *collapse_all_patterns=False*, *verbose=False*)

args_and_configs.**process_normalization_file**(*normalization_file*)

args_and_configs.**unique_attributes**(*patterns*)

8.2 etude.py Functions

etude.**align_tokens**(*reference_folder*, *test_folder*, *args*, *file_prefix='/'*, *file_suffix='.xml'*)
Align reference and test documents by token for comparison

etude.**collect_files**(*reference_folder*, *test_folder*, *file_prefix*, *file_suffix*, *skip_missing_files_flag*)

etude.**count_chars_profile**(*reference_ns*, *reference_dd*, *reference_folder*, *test_ns*, *test_dd*, *test_folder*, *args*, *file_prefix='/'*, *file_suffix='.xml'*)

etude.**count_ref_set**(*this_ns*, *this_dd*, *this_patterns*, *this_folder*, *args*, *file_prefix='/'*, *file_suffix='.xml'*, *set_type=None*)

etude.**create_output_folders**(*reference_out*, *test_out*)

etude.**generate_out_file**(*output_dir*, *input_filename*)
Generate a well-formed full file path for writing output stats

etude.**get_file_mapping**(*reference_folder*, *test_folder*, *file_prefix*, *file_suffix*, *skip_missing_files_flag*)

etude.**init_args**()

etude.**score_ref_set**(*reference_ns*, *reference_dd*, *reference_patterns*, *reference_folder*, *test_ns*, *test_dd*, *test_patterns*, *test_folder*, *args*, *file_prefix='/'*, *file_suffix='.xml'*)

8.3 scoring_metrics.py Functions

scoring_metrics.**accuracy**(*tp*, *fp*, *tn*, *fn*)

scoring_metrics.**add_missing_fields**(*score_summary*)

scoring_metrics.**document_level_annot_comparison_runner**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *fuzzy_flag*, *scorable_attributes*)

scoring_metrics.**end_comparison_runner**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *start_key*, *end_key*, *fuzzy_flag*, *scorable_attributes*, *scorable_engines*, *norm_synonyms*)

scoring_metrics.**evaluate_doc_properties**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_ss*, *test_ss*, *patterns*, *fuzzy_flag='doc-property'*, *scorable_attributes=[]*, *scorable_engines=[]*, *norm_synonyms={}*)

scoring_metrics.**evaluate_positions**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_ss*, *test_ss*, *fuzzy_flag='exact'*, *use_mapped_chars=False*, *scorable_attributes=[]*, *scorable_engines=[]*, *norm_synonyms={}*)

scoring_metrics.**exact_comparison_runner**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *start_key*, *end_key*, *fuzzy_flag*, *scorable_attributes*, *scorable_engines*, *norm_synonyms*)

scoring_metrics.**f_score**(*p*, *r*, *beta=1*)

scoring_metrics.**flatten_ss_dictionary**(*ss_dictionary*, *category='(unknown)'*)

scoring_metrics.**fully_contained_comparison_runner**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *start_key*, *end_key*, *fuzzy_flag*, *scorable_attributes*, *scorable_engines*, *norm_synonyms*)

scoring_metrics.**get_annotation_from_base_entry**(*annotation_entry*, *start_key*, *end_key*)

scoring_metrics.**get_unique_types**(*config*)

scoring_metrics.**new_score_card**(*fuzzy_flags=['exact'], normalization_engines=[]*)

scoring_metrics.**norm_summary**(*score_summary*, *args*)

scoring_metrics.**output_metrics**(*class_data*, *fuzzy_flag*, *metrics*, *delimiter_prefix*, *delimiter*, *stdout_flag*, *csv_out_filename*, *pretty_print_flag*)

scoring_metrics.**partial_comparison_runner**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *start_key*, *end_key*, *fuzzy_flag*, *scorable_attributes*, *scorable_engines*, *norm_synonyms*)

scoring_metrics.**precision**(*tp*, *fp*)

scoring_metrics.**print_2018_n2c2_track1**(*score_card*, *file_mapping*, *args*)

scoring_metrics.**print_confusion_matrix**(*confusion_matrix*, *file_mapping*, *reference_config*, *test_config*, *fuzzy_flag*, *args*)

scoring_metrics.**print_confusion_matrix_shell**(*confusion_matrix*, *file_mapping*, *reference_patterns*, *test_patterns*, *args*)

scoring_metrics.**print_counts_summary**(*score_card*, *file_list*, *config_patterns*, *args*, *set_type*)

scoring_metrics.**print_score_summary**(*score_card*, *file_mapping*, *reference_config*, *test_config*, *fuzzy_flag*, *args*, *norm_engine=''*)

scoring_metrics.**print_score_summary_shell**(*score_card*, *file_mapping*, *reference_config*, *test_config*, *args*)

scoring_metrics.**recall**(*tp*, *fn*)

scoring_metrics.**recursive_deep_key_value_pair**(*dictionary*, *path*, *key*, *value*)

scoring_metrics.**reference_annot_comparison_runner**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *start_key*, *end_key*, *fuzzy_flag*, *scorable_attributes*, *scorable_engines*, *norm_synonyms*)

scoring_metrics.**specificity**(*tn*, *fp*, *empty_value=None*)

scoring_metrics.**start_comparison_runner**(*reference_filename*, *confusion_matrix*, *score_card*, *reference_annot*, *test_entries*, *start_key*, *end_key*, *fuzzy_flag*, *scorable_attributes*, *scorable_engines*, *norm_synonyms*)

scoring_metrics.**update_confusion_matrix**(*confusion_matrix*, *fuzzy_flag*, *ref_type*, *test_type*)

scoring_metrics.**update_csv_output**(*csv_out_filename*, *delimiter*, *row_content*)

scoring_metrics.**update_output_dictionary**(*out_file*, *metric_type*, *metrics_keys*, *metrics_values*)

scoring_metrics.**update_score_card**(*condition*, *score_card*, *fuzzy_flag*, *filename*, *start_pos*, *end_pos*, *type*, *pivot_value=None*, *ref_annot=None*, *test_annot=None*, *scorable_attributes=None*, *scorable_engines=None*, *norm_synonyms={}*)

8.4 text_extraction.py Functions

text_extraction.**align_tokens_on_whitespace**(*dictionary*, *out_file*)

text_extraction.**create_annotation_entry**(*begin_pos=-1*, *begin_pos_mapped=None*, *end_pos=-1*, *end_pos_mapped=None*, *raw_text=None*, *pivot_attr=None*, *pivot_value=None*, *parity=None*, *tag_name=None*)

text_extraction.**extract_annotations**(*ingest_file*, *namespaces*, *document_data*, *patterns*, *skip_chars=None*, *out_file=None*)

text_extraction.**extract_annotations_brat_standoff**(*ingest_file*, *offset_mapping*, *type_prefix*, *tag_name*, *line_type*, *optional_attributes=[]*, *normalization_engines=[]*)

text_extraction.**extract_annotations_csv**(*csv_file*, *delimiter*, *tag_name*, *begin_column=None*, *end_column=None*, *text_column=None*, *optional_attributes=[]*)

text_extraction.**extract_annotations_json**(*ingest_file*, *raw_content*, *offset_mapping*, *annotation_path*, *tag_name*, *begin_attribute=None*, *end_attribute=None*, *optional_attributes=[]*, *normalization_engines=[]*)

text_extraction.**extract_annotations_plaintext**(*offset_mapping*, *raw_content*, *delimiter*, *tag_name*)

text_extraction.**extract_annotations_semeval_pipes**(*ingest_file*, *offset_mapping*, *tag_name*, *optional_attributes=[]*)

text_extraction.**extract_annotations_tsv**(*tsv_file*, *raw_content*, *offset_mapping*, *tag_name*, *optional_attributes=[]*)

text_extraction.**extract_annotations_xml**(*ingest_file*, *offset_mapping*, *annotation_path*, *tag_name*, *namespaces={}*, *begin_attribute=None*, *end_attribute=None*, *text_attribute=None*, *optional_attributes=[]*, *normalization_engines=[]*)

text_extraction.**extract_annotations_xml_spanless**(*ingest_file*, *annotation_path*, *tag_name*, *pivot_attribute*, *parity*, *namespaces={}*, *text_attribute=None*, *optional_attributes=[]*)

text_extraction.**extract_brat_attribute**(*ingest_file*, *annot_line*, *optional_attributes=[]*)

text_extraction.**extract_brat_equivalence**(*ingest_file*, *annot_line*, *optional_attributes=[]*)

text_extraction.**extract_brat_event**(*ingest_file*, *annot_line*, *tag_name*, *optional_attributes=[]*)

text_extraction.**extract_brat_normalization**(*ingest_file*, *annot_line*, *normalization_engines=[]*)

text_extraction.**extract_brat_relation**(*ingest_file*, *annot_line*, *tag_name*, *optional_attributes=[]*)

text_extraction.**extract_brat_text_bound_annotation**(*ingest_file*, *annot_line*, *offset_mapping*, *tag_name*, *line_type*, *optional_attributes=[]*)

text_extraction.**extract_chars**(*ingest_file*, *namespaces*, *document_data*, *skip_chars=None*)

text_extraction.**extract_json_chars**(*ingest_file*, *document_data*, *skip_chars=None*)

text_extraction.**extract_piped_text**(*ingest_file*, *skip_chars*)

text_extraction.**extract_plaintext**(*ingest_file*, *skip_chars*)

text_extraction.**map_position**(*offset_mapping*, *position*, *direction*)

Convert a character position to the closest non-skipped position.

Use the offset mapping dictionary to convert a position to the closest valid character position. We include a direction for the mapping because it is important to consider the closest position to the right or left of a position when mapping the start or end position, respectively.

Parameters

- **offset_mapping** – a dictionary mapping character positions to `None` if the character is in the skip list or to an int, otherwise
- **position** – current character position
- **direction** – 1, if moving right; -1 if moving left

Returns character position if all skipped characters were removed from the document and positions re-assigned or `None`, on KeyError

text_extraction.**split_content**(*raw_text*, *offset_mapping*, *skip_chars*)

text_extraction.**write_annotations_to_disk**(*annotations*, *out_file*)

CHAPTER 9

Configuration Files

9.1 Score Keys

- “Long Name” or “Section”
- “Short Name”
- custom

9.2 Score Values

CHAPTER 10

Input Formats & Support Details

10.1 Simple Plain Text

10.1.1 Newlines for Sentences

Local sample configuration files (under *config/*):

- *plaintext_sentences.conf*

10.2 Structured Plain Text (e.g., csv)

10.2.1 CSV with Start, End, and Negation Columns

- *csv_diagnoses.conf*

If the configuration file includes a key/value pair for *Opt Col*, then we forcibly include the following three available values for this column:

- affirmed
- negated
- possible

10.2.2 brat Annotation

The brat rapid annotation tool generates brat standoff format. Annotations are stored in a seconardy file (**.ann*) while the original text is found in a plain text file (**.txt*). This standoff format uses character offsets to locate spans: "*All offsets all [sic] indexed from 0 and include the character at the start offset but exclude the character at the end offset.*" See BioNLP Shared Task standoff format for a related format.

Limitations: The extraction engine currently only handles continous text-bound annotations for evaluation. Binary attributes can be extracted and included in the evaluation dictionary but are not scored themselves. Discontinous text-bound annotations, relations, events, multi-value attributes, normalizations, and notes are not supported.

Local sample configuration files (under *config/*):

- *brat_problems_allergies_standoff.conf*

10.3 XML Formats

10.3.1 UIMA CAS XMI

Local sample configuration files (under *config/*):

- *CAS_XMI.conf*
- *i2b2_2016_track-1.conf*
- *uima_sentences.conf*
- *webanno_phi_xmi.conf*
- *webanno_problems_allergies_xmi.conf*
- *webanno_uima_xmi.conf*

10.3.2 Other

Extra sample configuration files (via the ETUDE engine configs repository):

- *i2b2/...*
- *n2c2/n2c2_2018_track-1.conf*

CHAPTER 11

Evaluating Matches

Relevant command line arguments:

```
--fuzzy-match-flags exact
--fuzzy-match-flags partial
--fuzzy-match-flags fully-contained
--fuzzy-match-flags exact partial fully-contained

--fuzzy-match-flags [start|end|doc-property]

--ignore-whitespace
--skip-chars [regex of characters to skip]

--heed-whitespace
```

11.1 Exact Match

–fuzzy-match-flags exact

–fuzzy-match-flags exact partial fully-contained

11.2 Partial Match

–fuzzy-match-flags partial

11.3 Fully-Contained Match

–fuzzy-match-flags fully-contained

11.4 Start Match

–fuzzy-match-flags start

11.5 End Match

–fuzzy-match-flags end

11.6 Doc-Property Match

(In Development)

–fuzzy-match-flags doc-property

CHAPTER 12

Evaluating Normalization

Relevant command line arguments:

```
--score-normalization
--score-normalization <list,of,engines,to,evaluate>
--score-normalization <A/a,B/b>

--normalization-file <filename>
```

CHAPTER 13

Evaluating Context Attributes

Relevant command line arguments:

```
--score-attributes
--score-attributes <list,of,attributes,to,score>

--by-attribute
--by-type-and-attribute
```

13.1 Sample Test Data

Contextual attributes can be encoded as XML attributes. In the case of this sample WebAnno document, each attribute was a binary flag that could be set to *true* or *false*. True Negatives (TNs) are indicated by the corresponding XML attribute value being *False* (e.g., *negated="false"*).

```
python ${ETUDE_DIR}/etude.py \
   --reference-input ${ETUDE_DIR}/tests/data \
   --reference-config ${ETUDE_DIR}/config/webanno_problems_allergies_xmi.conf \
   --test-input ${ETUDE_DIR}/tests/data \
   --test-config ${ETUDE_DIR}/config/webanno_problems_allergies_xmi.conf \
   --file-suffix "013_Conditional_Problem.xmi" \
   --by-type \
   --by-attribute \
   --score-attributes
```

In the following brat *.ann* file, context attributes are similar to labels or notes added to an annotation. The default value should be made explicit. These files assume a concept is *Present* unless negated (e.g., *A17 Negated T22* is unpacked to meand that *A(nnotation)17* indicates that concept *T22* is *Negated*). Likewise, concepts are *Current* unless flagged *Historical*. True Negatives (TNs) are indicated by the absence of a label.

```
python3 ${ETUDE_DIR}/etude.py \
   --reference-input "${ETUDE_DIR}/tests/data/brat_reference" \
```

(continues on next page)

(continued from previous page)

```
   --reference-config "${ETUDE_DIR}/config/brat_problems_allergies_standoff.conf" \
   --test-input "${ETUDE_DIR}/tests/data/brat_system_out" \
   --test-config "${ETUDE_DIR}/config/brat_problems_allergies_standoff.conf" \
   --file-suffix ".ann" \
   --by-type \
   -m TP FP FN Precision Recall F1 \
   --by-attribute \
   --score-attributes \
   --by-type-and-attribute
```

This sample CSV file has a single column with multiple possible value in it. Because the metrics for *affirmed* vs. *negated* vs. *possible* are in direct conflict, it is not meaningful to compare their aggregate precision and recall. Instead, it is more reasonable to look at their relative performance or accuracy as an attribute. True Negatives (TNs) are indicated by a concept not being flagged for a particular label. Every TN counted for *affirmed* should correlate with a *TP*, *FP*, or *FN* with both *negated* and *possible*.

```
python3 etude.py \
   --reference-config config/csv_diagnoses.conf \
   --reference-input tests/data/csv_reference_out \
   --test-config config/csv_diagnoses.conf \
   --test-input tests/data/csv_system_out \
   --file-suffix ".csv" \
   --heed-whitespace \
   --fuzzy-match-flags exact partial \
   --score-attributes affirmed,negated,possible \
   --by-attribute \
   --by-type
```

CHAPTER 14

Output Formats

CHAPTER 15

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

args_and_configs, 23

e

etude, 24

s

scoring_metrics, 24

t

text_extraction, 26

Index

A

accuracy() (*in module scoring_metrics*), 24
add_missing_fields() (*in module scoring_metrics*), 24
align_patterns() (*in module args_and_configs*), 23
align_tokens() (*in module etude*), 24
align_tokens_on_whitespace() (*in module text_extraction*), 26
args_and_configs (*module*), 23

C

collect_files() (*in module etude*), 24
count_chars_profile() (*in module etude*), 24
count_ref_set() (*in module etude*), 24
create_annotation_entry() (*in module text_extraction*), 26
create_output_folders() (*in module etude*), 24

D

document_level_annot_comparison_runner() (*in module scoring_metrics*), 24

E

end_comparison_runner() (*in module scoring_metrics*), 24
etude (*module*), 24
evaluate_doc_properties() (*in module scoring_metrics*), 24
evaluate_positions() (*in module scoring_metrics*), 25
exact_comparison_runner() (*in module scoring_metrics*), 25
extract_annotations() (*in module text_extraction*), 26
extract_annotations_brat_standoff() (*in module text_extraction*), 26
extract_annotations_csv() (*in module text_extraction*), 26
extract_annotations_json() (*in module text_extraction*), 26
extract_annotations_plaintext() (*in module text_extraction*), 26
extract_annotations_semeval_pipes() (*in module text_extraction*), 26
extract_annotations_tsv() (*in module text_extraction*), 26
extract_annotations_xml() (*in module text_extraction*), 27
extract_annotations_xml_spanless() (*in module text_extraction*), 27
extract_brat_attribute() (*in module text_extraction*), 27
extract_brat_equivalence() (*in module text_extraction*), 27
extract_brat_event() (*in module text_extraction*), 27
extract_brat_normalization() (*in module text_extraction*), 27
extract_brat_patterns() (*in module args_and_configs*), 23
extract_brat_relation() (*in module text_extraction*), 27
extract_brat_text_bound_annotation() (*in module text_extraction*), 27
extract_chars() (*in module text_extraction*), 27
extract_delimited_patterns() (*in module args_and_configs*), 23
extract_document_data() (*in module args_and_configs*), 23
extract_json_chars() (*in module text_extraction*), 27
extract_json_patterns() (*in module args_and_configs*), 23
extract_namespaces() (*in module args_and_configs*), 23
extract_patterns() (*in module args_and_configs*), 23
extract_piped_text() (*in module*

text_extraction), 27
extract_plaintext() (*in module text_extraction*), 27
extract_semeval_patterns() (*in module args_and_configs*), 23
extract_xpath_patterns() (*in module args_and_configs*), 23
extract_xpath_spanless_patterns() (*in module args_and_configs*), 23

F

f_score() (*in module scoring_metrics*), 25
flatten_ss_dictionary() (*in module scoring_metrics*), 25
fully_contained_comparison_runner() (*in module scoring_metrics*), 25

G

generate_out_file() (*in module etude*), 24
get_annotation_from_base_entry() (*in module scoring_metrics*), 25
get_arguments() (*in module args_and_configs*), 24
get_file_mapping() (*in module etude*), 24
get_unique_types() (*in module scoring_metrics*), 25

I

init_args() (*in module etude*), 24
initialize_arg_parser() (*in module args_and_configs*), 24

M

map_position() (*in module text_extraction*), 27

N

new_score_card() (*in module scoring_metrics*), 25
norm_summary() (*in module scoring_metrics*), 25

O

output_metrics() (*in module scoring_metrics*), 25

P

partial_comparison_runner() (*in module scoring_metrics*), 25
precision() (*in module scoring_metrics*), 25
print_2018_n2c2_track1() (*in module scoring_metrics*), 25
print_confusion_matrix() (*in module scoring_metrics*), 25
print_confusion_matrix_shell() (*in module scoring_metrics*), 25
print_counts_summary() (*in module scoring_metrics*), 25
print_score_summary() (*in module scoring_metrics*), 25
print_score_summary_shell() (*in module scoring_metrics*), 25
process_config() (*in module args_and_configs*), 24
process_normalization_file() (*in module args_and_configs*), 24

R

recall() (*in module scoring_metrics*), 25
recursive_deep_key_value_pair() (*in module scoring_metrics*), 25
reference_annot_comparison_runner() (*in module scoring_metrics*), 25

S

score_ref_set() (*in module etude*), 24
scoring_metrics (*module*), 24
specificity() (*in module scoring_metrics*), 26
split_content() (*in module text_extraction*), 27
start_comparison_runner() (*in module scoring_metrics*), 26

T

text_extraction (*module*), 26

U

unique_attributes() (*in module args_and_configs*), 24
update_confusion_matrix() (*in module scoring_metrics*), 26
update_csv_output() (*in module scoring_metrics*), 26
update_output_dictionary() (*in module scoring_metrics*), 26
update_score_card() (*in module scoring_metrics*), 26

W

write_annotations_to_disk() (*in module text_extraction*), 27